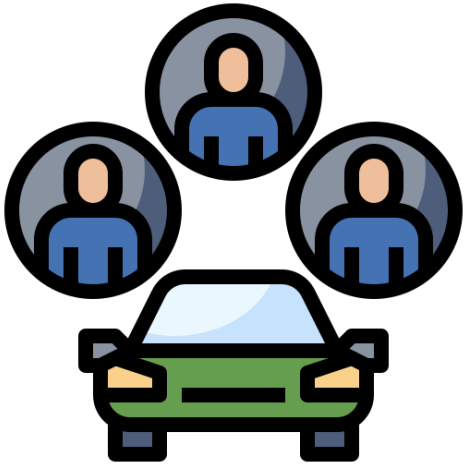
Semestrálny Project

Téma: Implementácia zdieľanej dopravy v C++ by Lukáš Hajdúch



Úvod

Vo vypracovanom projekte je možné vytvoriť jazdu používateľom a nastaviť start location, destination, cenu za 100 km a počet voľných miest zdieľanej jazdy. Ďalší užívateľ si môže zarezervovať, takúto jazdu a po dokončení za ňu aj zaplatiť a zanechať recenziu na používateľa, ktorý jazdu vytvoril. Informácie o jazde sa dajú zapísať do súboru a následne sa dajú z neho aj prečítať. Jazdy sa dajú vyhľadať podľa štartovej a cieľovej destinácie.

Ďalšia funkcionalita projektu je možnosť pridať jedno z 3 druhov vozidiel na prenájom: Electric car, Scooter a Electric bike. Používateľ si môže prenajať vozidlo a následne zaplatiť.

K dispozícii je taktiež výpis všetkých dostupných vozidiel alebo iba konkrétneho typu. Taktiež výpis jázd.

1: Trieda Review

```
class Review {
private:
    std::string comment; //Komentár k recenzii
    float rating; // Rating from 1 to 5

public:
    Review(const std::string& comment, float rating); //Konštruktor, ktorý vytvorí recenziu na
        //základe vstupných parametrov.
    const std::string& getComment() const; //Getter pre komentár.
    float getRating() const; //Getter pre hodnotenie.
};
```

2: Trieda User

```cpp
class User {
private:
    std::string username; //meno používateľa
    std::vector<Ride> rides; //vektor vytvorených jázd daného používateľa
    std::vector<Review> reviews; //vektor recenzií na daného používateľa
    bool reserved; //Má zarezervovanú jazdu, ak nie = False, inak True
    float money; //Peniaze používateľa
public:
    User(const std::string& name, float money1); //konštruktor nastaví
        //username a počet peňazí. Ak nie je name zadané priradí
        //RandomGeneratedUsername spoločne s číslom, podľa počtu randomne
        //vygenerovaných používateľov.
    bool addRide(const std::string& infoAboutRide);
        //format of infoAboutRide: <startingLocation>:<destination>:<pricePer100Km>:<routeLength>:<brandOfCar>:
        //<color>:<availableSeatsForReservation>
        //Táto funkcia rozdelí vstupný string a vytvorí inštanciu triedy Ride s parametrami zo stringu
        //infoAboutRide. Následne pridá danú jazdu "Ride" do vektora jázd daného používateľa. Vráti true, ak sú
        //všetky parametre korektne zadané, ak nejaký chýba vráti False.
    bool removeRide(const std::string& rideID);
        //Táto funkcia vymaže z vektora jázd daného používateľa jazdu s daným rideID.
        //Ak nájde takú jazdu vymaže ju a vráti True, inak vráti False.
    Ride findRideWithId(const std::string& rideID) const;
        //Daná funkcia prehľadá vektor jázd, pokiaľ vo vektore jázd nájde jazdu s
        //rideID, vráti danú jazdu. Ak nenájde vráti prázdnu jazdu Ride("-", "-","-", 0, 0, "-", "-", 0).
    bool addReview(const std::string& review);
        //format of review: "<number>:<comment>". Ak je recenzia zadaná v správnom
        //vytvorí inštanciu triedy Review s danými parametrami number (číselné desatinné hodnotenie = float) a
        //komentár. Následne pridá Review do vektora recenzií.
    float averageRating() const;
        //Vypočíta priemerné hodnotenie daného používateľa a vráti vypočítanú hodnotu. V prípade, ak nemá žiadne
        //hodnotenia funkcia vráti -1.00000.

}
```

2: Trieda User – pokračovanie

```
const std::string& getUsername() const;
    //Funkcia vráti členskú premennú username.
const std::vector<Ride> getRides() const;
    //Funkcia vráti vektor jázd daného používateľa.
std::string bestRating() const;
    //Funkcia vráti najlepšiu recenziu používateľa vo formáte "<number>:<comment>". Ak
    //používateľ nemá žiadne recenzie, funkcia vráti "No ratings for this user".
float getMoney() const { return money; } //Getter na počet peňazí daného používateľa.
void setMoney(float newAmount){ money = newAmount; } //Setter na počet peňazí používateľa.
bool getReservedRide() const{ return reserved; } //Getter na to, či má používateľ zarezerovanú
    //jazdu.
void setReservedRide(bool reserved1){ reserved = reserved1; } //Setter na rezerváciu jazdy.
```

3: Trieda Ride

```
class Ride {
private:
    std::string rideIdentificationNumber;
    std::string startingLocation;
    std::string destination;
    int pricePer100Km;
    int routeLength;
    std::string brandOfCar;
    std::string color;
    int availableSeatsForReservation;
    std::vector<std::string> idOfPassengers;

public:
    Ride(std::string& id, std::string& start, std::string& end, int price, int length, std::string& brandOfCar1, std::string& color1, int
        availableSeats); //Konštruktor Ride, nastaví všetky členské premenné podľa vstupných parametrov.
    const std::string& getStartingLocation() const; // Getter pre starting location.
    const std::string& getDestination() const; //Getter pre destination.
    int getPricePer100Km() const; //Getter pre cenu za 100 km.
    int getRouteLength() const; //Getter pre dĺžku cesty.
    const std::string& getBrandOfCar() const; //Getter pre značku auta.
    const std::string& getColor() const; //Getter pre farbu.
    int getAvailableSeatsForReservation() const; //Getter pre počet dostupných miest na sedenie.
    std::string getRideIdentificationNumber() const; //Getter pre rideID.
    std::string showInfoAboutRide() const; //Táto funkcia by mala vrátiť string vo Ride Ride ID: <RideId>\nStart location
        //<startingLocation>\nDestination: <destination>\nPrice per 100 km: <pricePer100Km>\nRoute length: <routeLength>\nBrand of car:
        //<brandOfCar>\n Color: <color>.
    void setAvailableSeatsForReservation(int newCountOfAvailableSeats){
        availableSeatsForReservation = newCountOfAvailableSeats;} // Prestavenie počtu dostupných miest na sedenie.
    void addIdOfPassengers(const std::string& idOfPassenger) {
        idOfPassengers.push_back(idOfPassenger); } // Pridanie pasažiera danej jazdy do vekora s idečkami pasažierov.
```

4: Trieda Vehicle

```cpp
enum TypesOfVehicles { SCOOTER, ELECTRIC_CAR, ELECTRIC_BIKE };
const std::string typesOfVehicles[] = {"Scooter", "Electric car", "Electric bike"}; //Konštantné pole typov vozdiel pre výpis

enum TypesOfBikes { TANDEM, CARGO, CLASSIC, KIDS };
const std::string typesOfBikes[] = {"Tandem", "Cargo", "Classic", "Kids"}; //Konštantné pole typov bicyklov pre výpis
class Vehicle {
protected:
    std::string identificationNumber; //ID of vehicle
    TypesOfVehicles typeOfVehicle; //Type of Vehicle from enum TypesOfVehicles
    float pricePerKilometer{}; //Rental price per kilometer
    bool available = true; //Is a vehicle available?
    int battery{}; //Battery 0 - 100

public:
    virtual ~Vehicle() = default;
    bool getAvailability() const; //Getter pre získanie, či je vozidlo available or not
    bool addPricePerKilometer(const std::string& price); //Vo funkcii sa treba pozrieť, či je správne zadaná cena. String price môže obsahovať iba
        //medzery, bodku alebo čísla od 0 po 9. Ak obsahuje niečo iné, vráti false, inak true.
    const std::string& getTypeofVehicle() const {
        return typesOfVehicles[typeOfVehicle];
    } //Funkcia vráti, akého typu je dané vozidlo. Vráti hodnotu z konštantného poľa, podľa členskej premmenej enum.
    float getPricePerKilometer() const {
        return pricePerKilometer;
    } //Getter pre prenajímateľnú cenu za 1 kilometer
    const std::string& getIdentificationNumber() const {
        return identificationNumber;
    } //Getter pre identifikačné číslo vozidla.
    int getBattery() const {
        return battery;
    } //Getter pre stav b
    void setBattery(int newBatteryLevel){
        if (available == false && newBatteryLevel > 10) {
            available = true;
        }
        battery = newBatteryLevel;
    } //Nastaví novú hodnotu batérie po prenájme, avšak ak bola teraz prenajatá a nový battery level by bol väčší ako 10. Nastav ju, ako dostupnú.
      //Inak nenastav, že je available, iba zmeň hodnotu battery level.
    void setAvailability(bool isAvailable){
        available = isAvailable;
    } //Nastav dostupnosť vozidla, podľa parametra isAvailable.
};
```

4: Trieda Vehicle – pokračovanie

```
class Scooter : public Vehicle {
public:
    Scooter() //Konštruktor pre podtriedu Scooter, ktorý nastaví všetky členské premenné, typeOfVehicle na hodnotu SCOOTER z enum TypesOfVehicles.
        //Battery nasatví na 100. Pridelí identifikačné číslo, ktoré sa bude po každom pridaní novej kolobežky zväčšovať. Formát identifikačného
        //čísla "KKXXX" - kde XXX je trojčíslie, pokiaľ je to prvá kolobežka, tak to bude "KK001".
    std::string showInfo() const; // Vypíše členské premenné identificationNumber, typeOfVehicle, battery, pricePerKilometer vo formáte:
        //Identification Number: <identificationNumber>\nType of vehicle: <typeOfVehicle>\nBattery: <battery>%\nPrice per Kilometer:
        //<pricePerKilometer>

};

class ElectricCar : public Vehicle {
private:
    bool airConditioning; //Klimatizácia
    int numberOfSeats; //Počet miest na sedenie v aute
    std::string model; //Model auta
    int acceleration; //Akcelerácia z 0 - 100 km/h
    int maximumRange; //Maximálny dojazd
    int topSpeed; //Najvyššia rýchlosť
    std::string spz; //ŠPZ
public:
    ElectricCar(const std::string& spz1, const std::string& model1, const int numberOfSeats1, const bool airConditioning1, const int acceleration1,
    const int topSpeed1, const int maximumRange1) //Konštruktor pre podtriedu ElectricCar, ktorý nastaví všetky členské premenné, typeOfVehicle na
         //hodnotu ELECTRIC_CAR z enum TypesOfVehicles. Battery nasatví na 100. Pridelí identifikačné číslo, ktoré sa po každom pridaní nového
         //elektrické auta bude zväčšovať. Formát identifikačného čísla "AAXXX" - kde XXX je trojčíslie, pokiaľ je to prvé elektrické auto, tak to
         //bude "AA001". Nastaví aj všetky členské premenné ElectricCar podľa vstupných parametrov konštruktora.
    const std::string& getIdentificationNumber() {
        return identificationNumber;
    } //Vráti identifikačné číslo vozidla.
    std::string showInfo() const; // Vypíše členské premenné identificationNumber, typeOfVehicle, battery, pricePerKilometer vo formáte:
        //Identification Number: <identificationNumber>\nType of vehicle: <typeOfVehicle>\nBattery: <battery>%\nPrice per Kilometer:
        //<pricePerKilometer>\nRegistration Number: <spz>\nAir Conditioning: ("YES" or "NO")\nNumber of Seats: <numberOfSeats>\nModel of Car:
        //<model>\nAcceleration (0-100): <acceleration> s\nMaximum Range: <maximumRange> km\nTop Speed: <topSpeed> km\h
    std::string getSPZ() const { return spz; }; //Getter na ŠPZ.
};

class ElectricBike : public Vehicle {
private:
    int typeOfBike; //Členská premenná typeOfBike
public:
    ElectricBike(const int typeOfBike1) //Konštruktor pre podtriedu Scooter, ktorý nastaví všetky členské premenné, typeOfVehicle na hodnotu
        //ELECTRIC_BIKE z enum TypesOfVehicles. Battery nasatví na 100. Pridelí identifikačné číslo, ktoré sa bude po každom pridaní nového e-biku
        //zväčšovať. Formát identifikačného čísla "BBXXX" - kde XXX je trojčíslie, pokiaľ je to prvý e-bike, tak to bude "BB001". Nastaví taktiež
        //členskú premennú ElectricBike s názvom typeOfBike podľa vstupného parametra konštruktora. Pokiaľ je číslo väčšie ako 3 (čo je veľkosť
        //enum TypesOfBike), tak vyhodí výnimku std::runtime_error("Failed to construct object."). Inak priradí typeOfBike = typeOfbike1.
    std::string showInfo() const;
    int getTypeOfBike() const { return typeOfBike; } //Getter pre hodnotu typeOfBike.
};
```

5: Trieda Manager

```
class Manager {
private:
    std::vector<User*> users; //Vektor smerníkov User, uchováva všetkých používateľov
    std::vector<ElectricCar*> electricCars; //Vektor smerníkov ElectricCar, uchováva všetky elektrické autá
    std::vector<ElectricBike*> electricBikes; //Vektor smerníkov ElectricBije, uchováva všetky elektrické bicykle
    std::vector<Scooter*> scooters; //Vektor smerníkov Scooter, uchováva všetky kolobežky

public:
    void addUser(User *user); //Pridá používateľa do vektora smerníkov User
    std::vector<Ride*> findRides(const std::string& source, const std::string& destination) const; //Prehľadá všetkých používateľov a pozrie sa, či
    //je dostupná jazda z source to destination, naplní nimi vektor smerníkov Ride a vráti ho. Budú tam všetky dostupné jazdy z bodu source do
    //destination.
    static std::string showInfoAboutRides(const std::vector<Ride*>& rides); //Ak je vektor smerníkov 'rides' prázdny, vráti "No rides from this
    //start location to that destination". V prípade, ak 'rides' nie sú prázdne vypíše informácie o jednotlivých jazdách v tvare <number>.
    //ride: ride->showInfoAboutRide()\n zavolá funkciu triedy Ride showInfoAboutRide() a takto pre každú jazdu z vektora rides.
    static bool reserveRide(User& user, Ride& ride); //Ak má user danú jazdu už zarezervovanú funkcia vráti false. Ak je v jazde ešte voľné miesto
    //a daný používateľ user nemá už zarezervovanú inú jazdu a má dostatok peňazí. Rezervácia prebehne úspešne, nastaví v jazde o jedno menej
    //miest na sedenie, pridá do id pasažiera do jatdy a nastaví danému používateľovi, že už má zarezervovanú platbu
    bool reserveVehicle(User& user, ElectricCar& vehicle, int distance) const; //Funkcia pozrie, či bude mať vozidlo dostatočnú batériu na
    //prejdenie 'distance', či nie je zarezervované niekým iným a skontroluje, či bude mať používateľ dostatok peňazí na prenajatie. Ak niečo
    //nie je splnené vráti false, inak zníži batériu o distance. Pokiaľ batéria klesne pod 10, tak nastaví availability na false a odčítata money.
    bool reserveVehicle(User& user, ElectricBike& vehicle, int distance) const;  //Similar as above.
    bool reserveVehicle(User& user, Scooter& vehicle, int distance) const; //Similar as above.
    void completePayment(User& user, Ride& ride) const; //Funkcia vykoná platbu za jazdu 'ride'. Odčíta peniaze a nastaví userovi, že nemá
        zarezervovanú jazdu.
    void addVehicle(Scooter* scooter1); //Pridá kolobežku do vektora smerníkov Scooter.
    void addVehicle(ElectricCar* electricCar1); //Pridá elektircké auto do vektora smerníkov ElectricCar.
    void addVehicle(ElectricBike* electricBike1); //Pridá elektrický bicykel do vektora smerníkov ElectricBike.
```

5: Trieda Manager – pokračovanie

```
std::string showAvailableAllVehicles() const; //Funkcia vypíše všetky dostupné vozidlá v danom formáte: Electric cars: AAXXX\n AAXXX\nScooters:
   //KKXXX\nElectric bikes: BBXXX\nBBXXX. Ak nie sú dostupné žiadne vozidlá vypíš "No vehicles available".
std::string showAvailableByType(const std::string& type) const; //Funkcia vypíše dostupné vozidlá iba konkrétneho typu, ak nie sú dostupné žiadne
   //vozidlá daného typu 'type', return "No vehicles with this type". Inak vypíše dostupné vozidlá daného 'type' vo formáte "BB005\nBB006". Iba ich
   //identifikačné čísla. Ak je zadaný zlý typ vozidla return "This type is not available".
bool deleteByType(const std::string& type); //Táto funkcia vymaže vektor s vozidlami daného typu 'type' a vr8ti true. Ak zadaný type nie je
   //dostupný return false.
void deleteAllVehicles(); //Vymaže všetky vektory elctricCars, scooters, electricBikes.
bool deleteByIdentificationNumber(const std::string& identificationNumber); // Vymaže vozidlo s daným ID 'identificationNumber' a vráti true. Ak
   //také vozidlo neexistuje vráti false.
void saveRideInfoIntoFile(User& user, Ride& ride); //Uloží informácie o ride do súboru, ktorý bude mať názov zložený z
   //user.username + ride.rideID.txt, vloží do neho informácie pomocou ride.showInfoAboutRide().
std::string readRideInfoFromFile(const std::string& nameOfFile); //Prečíta uložené údaje zo súboru, ak sa súbor nepodarí otvoriť vráti "File cannot
   //be opened.". Inak otvori súbor a prečíta súbor po riadkoch a vráti obsah súboru.
```

Ďakujem za pozornosť a verím, že sa vám
môj projekt páčil a splnil kritéria.
Prajem pekný zvyšok dňa.